*ARMY RESEARCH LABORATORY*

# Procedure for Substrate Temperature Control Using the Pyrometer During MBE Growth

Stefan Svensson

ARL-MR-491 October 2000

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

**Army Research Laboratory**

Adelphi, MD 20783-1197

**ARL-MR-491** **October 2000**

# Procedure for Substrate Temperature Control Using the Pyrometer During MBE Growth

Stefan Svensson

Sensors and Electron Devices Directorate

## Abstract

Command procedures have been developed for the Army Research Laboratory molecular beam epitaxy (MBE) computer control system that allow a user to automatically outgas and desorb the oxide from substrates before growth, as well as set substrate temperatures based on pyrometer readings during growths. These procedures allow completely unattended growth of structures once suitable temperatures have been determined.

# Contents



## Figure

## Background

The Army Research Laboratory molecular beam epitaxy (MBE) system is controlled by a PC-based system that uses the "Molly" software package supplied by EPI. The Molly system provides a script language that can be used to create command procedures that execute customized sequences of actions on the MBE system. Allowed actions are reading and setting cell temperatures, opening and closing shutters, reading pressure gauges, setting the azimuthal rotation speed of the substrate, and turning the substrate holder to the growth and load positions.

Before deposition of material is started, the native oxide on the substrate must be desorbed. This is done by heating the substrate to a predetermined temperature under As overpressure. Due to differences in emissivity of the various substrate holders because of design, size, and age, a specific thermocouple setting cannot be expected to produce the desired temperature with enough precision. Hence, small corrections to the set points must be applied, based on reading of the optical pyrometer.

Many device structures consisting of various material combinations may require different substrate temperatures during the growth sequence. For the reasons mentioned above, it is not possible to preprogram a temperature profile for the thermocouple since it may not result in the desired temperatures as read by the pyrometer. An operator must therefore be present to make the final temperature adjustments. Automation of these adjustments leads to time savings and, in some cases, greater precision and reproducibility.

## Script for Outgassing

The logic behind the script developed for substrate outgassing is as follows: The substrate temperature is first raised to a thermocouple set point, at and above which the pyrometer operates (440 °C). Below this temperature the bandgap of GaAs is large enough that the substrate is transparent, which results in a very high pyrometer reading since the pyrometer "sees" the heater elements behind the wafer. After the minimum operation temperature has been reached, the computer starts reading values from the pyrometer. (The critical temperature is considered reached when the pyrometer reading exceeds 450 °C *and* the time derivative of the temperature is positive.) The pyrometer can be read from either the analog or digital outputs of the instrument. The digital output is greatly preferred due to the

much lower noise level on the signal. (The analog output was used originally because of a malfunction with the digital output port and produced a signal that required heavy averaging and thus more script code.)

The ramp between the minimum operating temperature of the pyrometer and the final outgassing temperature is approximated with a 10-step staircase during which a PID-control (proportional integration and differential) routine drives the pyrometer temperature to each set point. When the final outgassing temperature is reached, it is held constant with software PID control for a specific time, typically 10 min. After this, the temperature is ramped down to a desired growth temperature with the same control algorithm. The outgassing script has now finished and a growth script can be started. The code for the outgassing sequence is called **Genoutg.cmd** and can be used for both GaAs and InP substrates. A parameter flag must be set to provide information about which substrate is used. The control code for the outgassing script is shown in in the appendix (Command File Listings).

## Script for Temperature Control During Growth

By running the script **Tsub.cmd** in parallel with the structure recipe during growth, one can change the substrate temperature, as read by the pyrometer, from the structure recipe. Again, PID software control is used. The temperature set point is stored in a global variable (see next section), which is read every 5 s by **Tsub.cmd** and can be changed in the recipe by loading of command files named **TsubXXX.cmd**, where XXX is the temperature. Allowed values are 450 °C < XXX < 700 °C in 5° increments.

## Starting the Scripts

The appendix shows an example of a structure recipe for a PHEMT (pseudomophic high-electron-mobility transistor), which will be used to illustrate the start of the scripts. All command procedures are invoked with the Molly command *load*. A few comments about this command are appropriate at this point. The Molly language also includes a command *nsh*, which can "launch" another command procedure. The difference between loading and launching is that a loaded command file must execute to completion before the next step in the calling (loading) file can continue. A launch, on the other hand, starts the new command file and immediately continues with the next step. In other words, the calling and the called procedure will run in parallel. For unknown reasons, the Molly spreadsheet accepts only

the *load* command and not *nsh.* Incidentally, the new EPICAD program allows both *load* and *nsh*.

The first command file, **Stdlog1.cmd** (see appendix), starts a data logger. This is a personal preference of mine and is not needed for the pyrometer control. The same is true for the next command, **Rot5.cmd**, which sets the rotation speed. Next, the outgassing procedure, **Genoutg.cmd**, follows. This should run to completion before the growth begins and is therefore started directly with a *load.*

Since we clearly want to allow the temperature controller and the following structure recipe file to run in parallel, we must load an additional intermediate file, **Pyroc.cmd** (see appendix), which contains only the *nsh* command that launches the desired file (**Tsub.cmd**) (see appendix). This is done immediately after the outgassing.

The temperature set point, which is used in the PID control, is the same as the last temperature in **Genoutg.cmd.** This temperature is stored in the global variable *Tsub_new*, which is declared in the file **C:\molly\lib\local.cmd,** a command file that is executed automatically when the Molly system is started. One can make subsequent changes to the substrate temperature by loading files called **TsubXXX.cmd**, where XXX is the temperature (see appendix). Allowed values are 450 °C < XXX < 700 °C in 5° increments. These command files only set the global variable to a new value, so their execution time is negligible. If a different temperature is desired, any of the existing files can be copied and renamed to the new temperature and the set point edited. (Temperatures below 450 °C should not be used for the obvious reason that the pyrometer does not work in that range.) In the PHEMT recipe, loading **Tsub500.cmd** lowers the temperature from 600 °C, which is the temperature at the finish of the outgassing, to 500 °C just before the InGaAs layer is started.

Before the InGaAs growth starts, another GaAs layer is inserted after the **Tsub500.cmd.** This illustrates an important limitation of the PID routine. If the temperature change is too large, an abrupt change in set point may result in a substantial under- or overshoot. An undershoot can be particularly problematic since it may drive the substrate temperature below the operating range of the pyrometer. To avoid this, the **Tsub.cmd** file always applies a ramp with a rate of 40°/min. Hence, in the PHEMT example, the new set point, which is 100 °C below the previous one, will not be reached until 150 s after **Tsub500.cmd** is loaded, requiring the extra GaAs layer (180 s is used in the recipe for extra margin).

When the recipe has finished, the **Tsub.cmd** should be terminated. This is done by loading **Kill_pyr.cmd** (appendix). The latter executes the command *kill(pyro_id, SIGKILL)* in which *pyro_id* is another global variable defined in **C:\molly\lib\local.cmd**. Similarly, the logging is stopped with the file **Kill_log.cmd** (appendix). The files **Rot0.cmd** and **Pos1.cmd** turn off the azimuthal rotation and turn the substrate to the loading position. (Note

that the pyrometer control definitely *must* be turned off before the substrate is turned in another direction.)

Figure 1 shows the results of two consecutive growths (run numbers 1281 and 1282) with the automated outgassing and the change of substrate temperature during deposition. Despite the fact that two different types of substrate holders (EPI UNI-Block and Varian nonbonded) were used, the pyrometer traces are virtually identical beyond the outgassing phase.

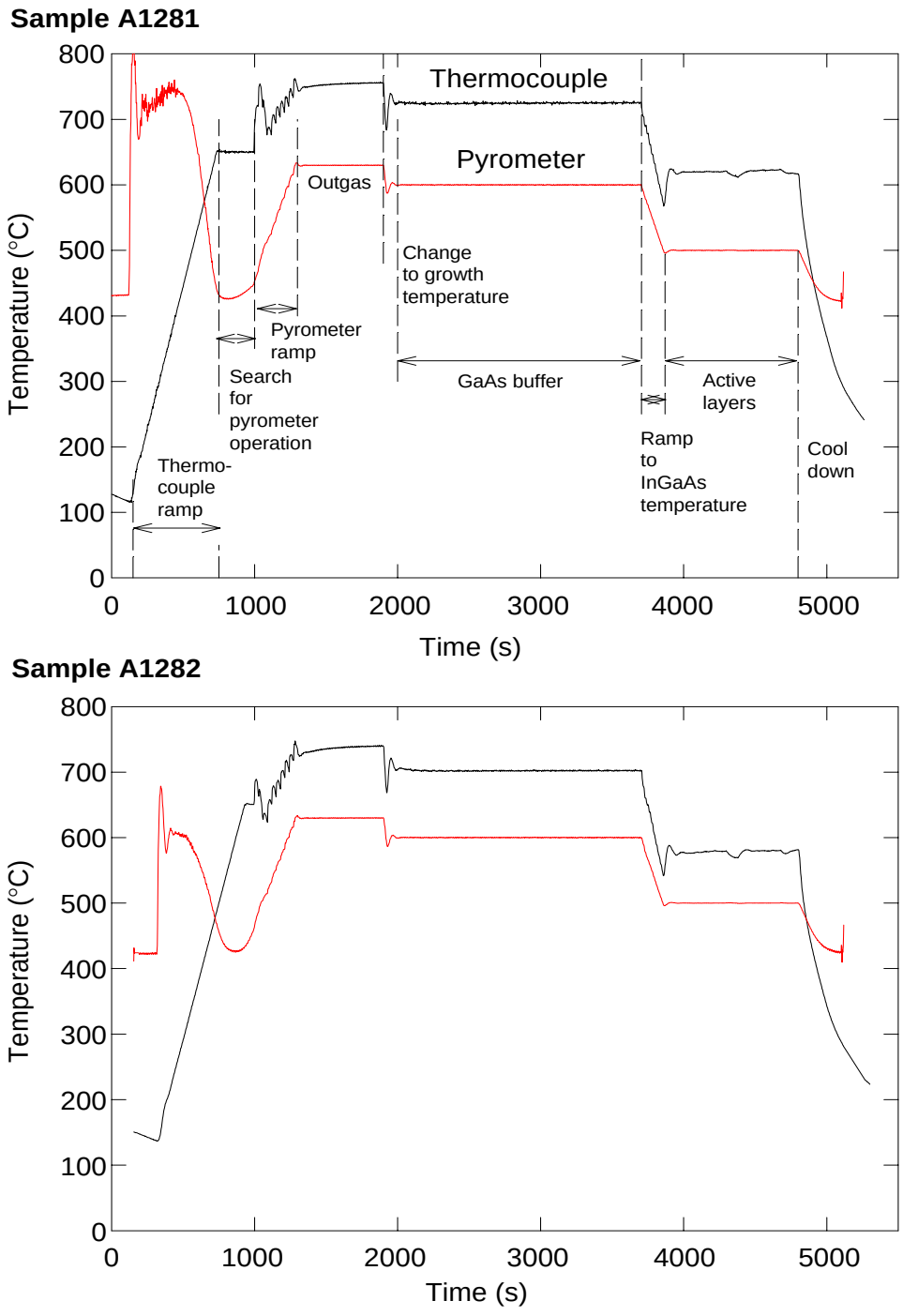


Figure 1. Temperature as a function of time during growth of two consecutive PHEMT wafers. Solid trace represents thermocouple data and dotted trace represents pyrometer temperature.

## Final Comments

The files **Genoutgas.cmd, Tsub.cmd, TsubXXX.cmd,** and **Pyroc.cmd** are all stored in **C:\u\mbe\.** This allows the use of the file name only (without the explicit path name) in the growth recipe. As mentioned above, the global parameters are created in **C:\molly\lib\local.cmd.**

The parameters that are coded into the script files all have comments next to their declarations. It should be fairly obvious from the script listings what the parameters represent. Some optimization has been done and the present values do work. However, further optimization can probably be done and certainly refinements are possible. In particular, I am planning to allow a step change in temperature if the difference between the old and new set points is below a certain value. Once I am satisfied that the outgassing routine works perfectly for both InP and GaAs, I will probably create separate files so that the operator does not have to edit the parameter *SubToday.*

## Acknowledgment

I used the PID controller, coded and optimized by Rich Leavitt.

## Appendix. Command File Listings

| | |
|---|---|
| **PHEMT8.wb1** | Molly spreadsheet recipe for a PHEMT |
| **Genoutgas.cmd** | General outgassing procedure for GaAs and InP |
| **Stdlog1.cmd** | Starts the standard logging procedure |
| **Pyroc.cmd** | Launches **Tsub.cmd** |
| **Tsub.cmd** | General PID controller for MBE growth |
| **Tsub550.cmd** | Changes the pyrometer set point to 550 °C |
| **Kill_pyr.cmd** | Terminates **Tsub.cmd** |
| **Kill_log.cmd** | Terminates the data logging |

| | |
|---|---|
| **Title:** | phemt8 |
| **Description:** | ARL PHEMT-8 |
| **Substrate:** | GaAs |
| **Lattice Constant:** | 5.6533 |

**EDIT FRAMED PARAMETERS**

| | | | |
|---|---|---|---|
| **StdGrow:** | 1 um/hr | = | 2.7778 A/sec |
| **StdFlux:** | 6.149E+14 /cm2-sec | | |

| | |
|---|---|
| | **Defaults:** |
| **Duration:** | time |
| **Cell Output:** | flux |
| | **Units** |
| **Thickness:** | Angstrom |
| **Time:** | sec |
| **Temperature:** | C |

Ox des 1
Cell cool 0
Si pause 5 min

Down rmp1 5 min
Down rmp2 5 min

Table

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | old | | |
| Si | -2.2815 | 32.691 | | -2.282 | 32.69 | -2.31 | 32.8 |

hall cal 1/29/99

| ND | T Si | Ga mu/h | Mole frac |
|---|---|---|---|
| 5.5E+17 | 1264.697 | 1 | 0.23 |
| 4E+18 | 1346.355 | 1 | 0.23 |

500 Max Si ramprate (deg/min)

DON"T DELET THIS COLUMN
DON"T DELET THIS COLUMN

**Comments:**
PHEMT with pyrometer control

| StartRecipe | | | Ga | | | Al3 | | | Al4 | | | In | | | Si | | | Be | | | As | | | As_valve | | | Sb | | | subs | | | EndCells | Thickn (A) | Doping | Mole fr | Gr rate |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Label** | **Action** | **Object** | x | R | S | x | R | S | x | R | S | x | R | S | T | R | S | T | R | S | T | R | S | T | R | S | T | R | S | T | R | | | | | | |
| Start_log | load | Stdlog1.cmd | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Turn_on_rotation | load | Rot5.cmd | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Desorb | load | Genoutg.cmd | | | | | | | | | | | | | | | | | | | X | | | | | | | | | | | | 1 | | | | |
| Start_pyro_contrl | load | pyroc.cmd | | | | | | | | | | | | | | | | | | | X | | | | | | | | | | | | | | | | |
| GaAs_Buffer_600C | grow | 1620.00 | | X | | | | | | | | | | | 1264.697 | 200 | | | | | X | | | | | | | | | | | | 4500 | | | 1 | |
| GaAs_Buffer_ramp | load | Tsub500.cmd | | X | | | | | | | | | | | | | | | | | X | | | | | | | | | | | | 0 | | | 1 | |
| GaAs_Buffer_500C | grow | 180.00 | | X | | | | | | | | | | | | | | | | | X | | | | | | | | | | | | 500 | | | 1 | |
| InGaAs_Channel | grow | 45.90 | | X | | | | | | | | X | | | | | | | | | X | | | | | | | | | | | | 150 | | 15 | 1.17647 | |
| AlGaAs_Spacer | grow | 13.86 | | X | | | | | X | | | | | | | | | | | | X | | | | | | | | | | | | 50 | | 23 | 1.2987 | |
| Si_Pulse | pause | 352.80 | | | | | | | | | | | | | | | X | | | | X | | | | | | | | | | | | | 7.00E+12 | | 1 | |
| AlGaAs_Top | grow | 83.16 | | X | | | | | X | | | | | | | | X | | | | X | | | | | | | | | | | | 300 | 5.5E+17 | 23 | 1.2987 | |
| Si_up_ramp | pause | 300.00 | | | | | | | | | | | | | 1346.355 | 500 | | | | | X | | | | | | | | | | | | | | | | |
| GaAs_Cap | grow | 126.00 | | X | | | | | | | | | | | | | X | | | | X | | | | | | | | | | | | 350 | 4.00E+18 | | 1 | |
| Stop_pyro_ctrl | load | Kill_pyr.cmd | | | | | | | | | | | | | | | | | | | X | | | | | | | | | | | | | | | | |
| Ramp_down_Tsub | pause | 300.00 | | | | | | | | | | | | | | | | | | | X | | | | | | | | | | 25 | 200 | | | | | |
| Turn_off_rotation | load | Rot0.cmd | | | | | | | | | | | | | | | | | | | X | | | | | | | | | | | | | | | | |
| Turn_wafer_to_load | load | Pos1.cmd | | | | | | | | | | | | | | | | | | | X | | | | | | | | | | | | | | | | |
| Close_As_Ramp_Tsub | pause | 300.00 | | | | | | | | | | | | | 1264.697 | 200 | | | | | | | | 0 | 200 | | | | | | | | | | | | |
| Ramp_down_cells | pause | 0.00 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Stop_log | load | Kill_log.cmd | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | sec | 3321.72 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | min | 55.36 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Hr | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | +Min | 55.362 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**24-Jul-2000 phemt8**

**Auto oxide desob - Variable**

**Growth**

**Cells ramp down - Variable**

```
/**/
/*****************************************************************************************/
/*                                                                                       */
/*   This command file desorbs the oxide from an substrate which can be either          */
/*   GaAs or InP (not tested yet)                                                        */
/*   Temperature control via pyrometer and PID routine from Rich Leavitt                 */
/*                                                                                       */
/*                                    Stefan Svensson, ARL  Mar 9, 2000                  */
/*****************************************************************************************/
/**/
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>
#include <cells.h>
#include <gauges.h>
#include <shutters.h>
#include <time.h>
#include <mbe.h>
/**/
/*=======================================================================================*/
/*====================== Declarations and initial values ================================*/
/*=======================================================================================*/
/**/
/*                                                                                       */
/*                        /---------\  -TPoutgas                                         */
/*                       /           \                                                   */
/*                      /             \                                                  */
/*       TC_work-    /-/               \------ TPGrow                                    */
/*                  /                                                                    */
/*                 /                                                                     */
/*                /                                                                      */
/*               /                                                                       */
/*            ___/                                                                       */
/*               |---|  |--|        |--|                                                */
/*         Ramp_time1  Ramp_time2  Ramp_time3                                            */
/*               |--|     |---------|                                                    */
/*            Wait_time1  Outg_time                                                      */
/*                 | Pyro reading starts                                                 */
/**/
double Substrate;             /* Substrate flag                                          */
double GaAs         = 1;      /* GaAs                                                    */
double InP          = 0;      /* InP                                                     */
double SubToday     = 1;      /* Today's substrate                                       */
double TPoutgasInP  = 550.0;  /* Pyrometer    Max outgassing temp  InP                   */
double TPoutgasGaAs = 630.0;  /* Pyrometer    Max outgassing temp  GaAs                  */
double TPoutgas;              /* Pyrometer    Max outgassing temp  General               */
double TP_growInP   = 525.0;  /* Pyrometer    Growth temperature InP                     */
double TP_growGaAs  = 600.0;  /* Pyrometer    Growth temperature GaAs                    */
double TP_grow;               /* Pyrometer    Growth temperature General                 */
double TPdelta;               /* Pyrometer    Difference between last two readings       */
double TPlast;                /* Pyrometer    Previous reading                           */
double TPsub;                 /* Pyrometer    general substrate temperature              */
double TP0;                   /* Pyrometer    reading at start of PID control ramp       */
double TPwork       = 440.0;  /* Pyrometer    min real temperature                       */
double TPstep;                /* Pyrometer    step length in PID control ramp            */
double TCsub;                 /* Thermocouple general substrate temperature              */
double TC_work;               /* Thermocouple Min operating temp (first ramp target)     */
double TC_workGaAs  = 650.0;  /* Thermocouple Min operating temp GaAs                    */
double TC_workInP   = 500.0;  /* Thermocouple Min operating temp InP                     */
double TC_test      = 450.0;  /* Thermocouple start testing for pyrometer operation      */
double Ramp_rate;             /* Thermocouple Ramp rate                                  */
double Ramp_time1   = 10.0;   /* Minutes      ramp time to temp where pyrometer works    */
double Ramp_time2   =   5.0;  /* Minutes      ramp time to outgassing temp               */
double Ramp_time3   =   3.0;  /* Minutes      ramp time to growth temp                   */
double Wait_time1;            /* Minutes      to wait for pyrometer to start working     */
double Wait_time2   =   0.1;  /* Minutes      to wait between pyrometer readings         */
double Wait_time3   =   0.5;  /* Minutes      extra wait before pyrometer control        */
```

```
double Outg_time    =  10.0;  /* Minutes      outgas time                              */
double Time_step;             /* Time step length in ramp                              */
double Numb_steps   =  10.0;  /* Number of steps that ramp is broken into              */
int    i;                     /* Counter for pyrometer readings                        */
int    I_pyro       =  5;     /* Number of pyrometer readings to average               */

double
  t0,
  control_time,
  Tstep = 1,
  sigma,
  delta,
  Prop = 72,
  Integ = 4.7,
  Deriv = 1.35,
  Tmin = 25,
  Tmax = 850,
  errtot=0,
  TCsetp,
  oldT,
  newT,
  curr_err,
  prev_err;
string sznewT,szct;
/**/
/*=============================================================================================*/
/*============= Initialize parameters for the right substrate ==========================*/
/*=============================================================================================*/
/**/
if(SubToday == GaAs)
   {
    TPoutgas = TPoutgasGaAs;
    TP_grow  = TP_growGaAs;
    TC_work  = TC_workGaAs;
    }
if(SubToday == InP)
   {
    TPoutgas = TPoutgasInP;
    TP_grow  = TP_growInP;
    TC_work  = TC_workInP;
    }
echo(“ TPoutgas = “,TPoutgas);
echo(“ TP_grow  = “,TP_grow);
echo(“ TC_work  = “,TC_work);
/**/
/*=============================================================================================*/
/*======================== Ramp substrate to the range where  ============================*/
/*======================== the pyrometer starts working       ============================*/
/*=============================================================================================*/
/**/
while( TCsub <= 0.0 ) TCsub = temp( subs );         /*  Read current TC set point      */
Ramp_rate = fabs( (TC_work  - TCsub )/Ramp_time1);
set_ramp( subs ,Ramp_rate );                         /*  Ramp thermocouple              */
set_temp( subs ,TC_work);                            /*        to temp where pyro works */
echo(“”);
echo(“ Ramping substrate to pyrometer range (Thermo-couple= “,TC_work,” )”);
echo(“ This will take “,Ramp_time1,” min.”   );
echo(“”);                                            /*  Wait until ramp is done        */
echo(“ Waiting for substrate to thermalize and pyrometer to start working “);
Wait_time1 =  Ramp_time1*1.1;
sleep ( Wait_time1*60);                              /*  Start checking pyro just before*/
/*                                                       ramp finsihes                  */
```

```
/*=============================================================================================*/
/*=========== Start testing for pyrometer operation ================================== */
/*=============================================================================================*/
/**/
echo(“ Looking for pyrometer stability “);
TPdelta =   -1.0;                                   /* initialize                       */
TPlast  = 1000.0;                                   /* Search for positive TPdelta      */
TCsub   =    0.0;
TPsub   =    0.0;
while (TPdelta <0.0 || TPsub < TPwork)
    {
    i=0;
    TPsub = 0.0;
    while (i < I_pyro)
        {
        i=i+1;
        TPsub = TPsub + reading(pyrometer);
/*                  echo(“ i, TPsub =”,i,TPsub);   */
        sleep(0.5);
        }
    TPsub   = TPsub/I_pyro;
    TPdelta = TPsub - TPlast;
    TPlast  = TPsub;
/*              echo(“ TPsub, TPdelta = “, TPsub, TPdelta );   */
    sleep ( Wait_time2*60);
    }
sleep ( Wait_time3*60);                             /* Wait an extra 30 sec (chicken!)  */
/**/
/*=============================================================================================*/
/*====================== Ramp to outgas temp  =================================================*/
/*=============================================================================================*/
/**/
echo(“ Starting pyrometer control and upramp”);
echo(“”);
Time_step    = Ramp_time2*60.0/Numb_steps;                  /* Time per step in sec    */
control_time = Time_step;
TP0          = reading(pyrometer);
TPstep       = (TPoutgas - TP0)/Numb_steps;
echo(“ Current temperature = “,TP0);
/* echo(“control_time = “,control_time);      */
/* echo(“TPstep  = “,TPstep );                */
/* echo(“Tstep  = “,Tstep );                  */
i            = 0;
while (i < Numb_steps)
      {
      i        = i+1;
      newT     = TPstep*i + TP0;
      echo(“New temp:”,newT);
      sigma    = -setp(subs)*Integ*Prop/100;
      prev_err = reading(pyrometer)-newT;
      set_ramp(subs,1000);
      oldT     = newT;
      tm       = t;
      t0       = t;
      while ( t-t0 < control_time )
            {
            sleep(Tstep - (t - tm));
            curr_err = reading(pyrometer)-newT;
            sigma    = sigma+Tstep*curr_err;
            errtot   = errtot+Tstep*curr_err*curr_err;
            delta    = (curr_err-prev_err)/Tstep;
            TCsetp   = -(curr_err+sigma/Integ+delta*Deriv)*100/Prop;
            if (TCsetp < Tmin) TCsetp = Tmin;
            if (TCsetp > Tmax) TCsetp = Tmax;
            set_temp(subs,TCsetp);
            prev_err = curr_err;
```

```
            tm = tm + Tstep;
            tm=t;
            }
       }
/**/
/*=============================================================================*/
/*======================  Outgas ===============================================*/
/*=============================================================================*/
/**/
echo(“ Outgas temperature reached “);
echo(“ Outgassing for “, Outg_time, “ min”);
/**/
      control_time = Outg_time*60.0;
      newT         = TPoutgas;
/*      echo(“New temp:”,newT);     */
      sigma        = -setp(subs)*Integ*Prop/100;
      prev_err=reading(pyrometer)-newT;
      set_ramp(subs,1000);
      oldT         = newT;
      tm           = t;
      t0           = t;
      while ( t-t0 < control_time )
            {
            sleep(Tstep - (t - tm));
            curr_err = reading(pyrometer)-newT;
            sigma    = sigma+Tstep*curr_err;
            errtot   = errtot+Tstep*curr_err*curr_err;
            delta    = (curr_err-prev_err)/Tstep;
            TCsetp   = -(curr_err+sigma/Integ+delta*Deriv)*100/Prop;
            if (TCsetp < Tmin) TCsetp = Tmin;
            if (TCsetp > Tmax) TCsetp = Tmax;
            set_temp(subs,TCsetp);
            prev_err = curr_err;
            tm       = tm + Tstep;
            tm       = t;
            }
/**/
/*=============================================================================*/
/*====================== Ramp to growth temp  =================================*/
/*=============================================================================*/
/**/
echo(“”);
echo(“ Done outgassing “);
echo(“ Starting “,Ramp_time3,” min down-ramp”);
echo(“”);
control_time = Ramp_time3*60;
TP0          = reading(pyrometer);
echo(“Current temperature = “,TP0);

      newT     = TP_grow;
      echo(“New temp:”,newT);
      sigma    = -setp(subs)*Integ*Prop/100;
      prev_err = reading(pyrometer)-newT;
      set_ramp(subs,1000);
      oldT     = newT;
      tm       = t;
      t0       = t;
      while ( t-t0 < control_time )
            {
            sleep(Tstep - (t - tm));
            curr_err = reading(pyrometer)-newT;
            sigma    = sigma+Tstep*curr_err;
            errtot   = errtot+Tstep*curr_err*curr_err;
            delta    = (curr_err-prev_err)/Tstep;
            TCsetp   = -(curr_err+sigma/Integ+delta*Deriv)*100/Prop;
            if (TCsetp < Tmin) TCsetp = Tmin;
            if (TCsetp > Tmax) TCsetp = Tmax;
```

```
            set_temp(subs,TCsetp);
            prev_err = curr_err;
            tm       = tm + Tstep;
            tm       = t;
            }
/**/
/*=============================================================================*/
/*======================Ready to grow ==========================================*/
/*=============================================================================*/
/**/
Tsub_new = TP_grow;
echo(“ Substrate outgassed and oxide desorbed “);
echo(“ Ready for growth”);
```

```
/*************************************************************************/
/*                                                                       */
/*  This command procedure starts a data log of temperatures,            */
/*  shutter positions and the flux gauge during a growth.                */
/*  The data is stored in the file                                       */
/*  C:/u/mbe/recipes/stefan/log_data/outfile.dat                         */
/*                                                                       */
/*************************************************************************/

log_id = logger(20.0,                        /* log every 20 seconds */
            't',
            'temp(subs)',
            'temp(Ga)',
            'temp(Al3)',
            'temp(Al4)',
            'temp(In)',
            'temp(Si)',
            'temp(Be)',
            'temp(Sb)',
            'temp(As)',
            'temp(As_valve)',
            'is_open(Ga)',
            'is_open(Al3)',
            'is_open(Al4)',
            'is_open(In)',
            'is_open(Si)',
            'is_open(Be)',
            'is_open(Sb)',
            'is_open(As)',
            'reading(flux)',
            "c:///u//mbe//recipes//stefan//log_data//outfile.dat");
echo("log_id = ",log_id);
```

```
/**/
/*****************************************************************************/
/*                                                                           */
/*   This command file starts the pyrometer control program Tsub.cmd         */
/*   By launcing it instead of loading it, it will run in parallel with      */
/*   the rest of the growth.                                                 */
/*                                    Stefan Svensson, ARL  May 24 2000      */
/*****************************************************************************/
/**/
/**/
pyro_id = nsh(“Tsub.cmd”);      /* Process ID of the launch file */
echo(“ Launched  Tsub.cmd”);
```

```
/**/
/*****************************************************************************/
/*                                                                           */
/*   This command file sets a new substrate temperature over a time interval of       */
/*   50000 seconds (in other words it runs until you kill it).                        */
/*   New setpoints are entered by setting the global variable Tsub_new in an other    */
/*   command file. Enter the line load Tsubxxx.cmd (where xxx is the temp)            */
/*   Temperature control via pyrometer and PID routine from Rich Leavitt              */
/*                                                                           */
/*                                    Stefan Svensson, ARL  Jun 09, 2000             */
/*****************************************************************************/
/**/
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>
#include <cells.h>
#include <gauges.h>
#include <shutters.h>
#include <time.h>
#include <mbe.h>
/**/
/*===========================================================================*/
/*====================== Declarations and initial values ====================*/
/*===========================================================================*/
/**/
double recipe_time  = 50000.0;/* Seconds total time the cmd file is running              */
double t00;                   /* Seconds start time for cmd file                         */
double TP0;                   /* Pyrometer    reading at start of PID control ramp       */
double TPstep;                /* Pyrometer    step length in PID control ramp            */
double TCsub;                 /* Thermocouple general substrate temperature              */
double TC_test      = 450.0;  /* Thermocouple start testing for pyrometer operation      */
double Ramp_rate    = 40;     /* Thermocouple Ramp rate  (deg/min)                       */
double Time_step;             /* Time step length in ramp                                */
double Numb_steps   =  10.0;  /* Number of steps that ramp is broken into                */
int    i;                     /* Counter for pyrometer readings                          */
int    I_pyro       =  5;     /* Number of pyrometer readings to average                 */
double newT;                  /* Pyrometer  New temperature value used inside PID loop   */
double t0;                    /* Seconds    Start time for PID loop                      */
double control_time;          /* Seconds    duration of PID loop                         */
double Tstep        = 1.0;    /* Seconds    wait time between PID steps                  */
double sigma;                 /* PID sigma                                               */
double delta;                 /* PID delta                                               */
double Prop         = 72.0;   /* PID prop constant                                       */
double Integ        = 4.7;    /* PID integration constant                                */
double Deriv        = 1.35;   /* PID derivative constant                                 */
double TCmin        = 25.0;   /* Thermocouple min allowed value                          */
double TCmax        = 850.0;  /* Thermocouple min allowed value                          */
double TCsetp;                /* Thermocouple setpoint                                   */
double errtot       = 0.0;    /* PID total error                                         */
double curr_err;              /* PID current error                                       */
double prev_err;              /* PID previous error                                      */
/**/
/*===========================================================================*/
/*====================== Check for setpoint changes =========================*/
/*===========================================================================*/
/**/
t00       = t;                          /*   Start time  */
while ( t-t00 < recipe_time )
      {

/*      echo("First test line, Tsub_new = ",Tsub_new);    */

      if( Tsub_old != Tsub_new )
         {
/**/
/*===========================================================================*/
/*====================== Ramp to new temp  ==================================*/
```

```
/*=============================================================================================*/
/**/
         echo(“ Starting pyrometer control to “,Tsub_new );
         echo(“”);
         TP0          = reading(pyrometer);
         TPstep       = (Tsub_new - TP0)/Numb_steps;
         Time_step    = fabs(TPstep)/Ramp_rate*60.0;     /* Time per step in sec    */
         control_time = Time_step;
         i            = 0;
/*         echo(“ Current temperature = “,TP0);  */
/**/
         while (i < Numb_steps)
               {
               i        = i+1;
               newT     = TPstep*i + TP0;
                                                 echo(“New temp:”,newT);
               sigma    = -setp(subs)*Integ*Prop/100;
               prev_err = reading(pyrometer)-newT;
               set_ramp(subs,1000);
               tm       = t;
               t0       = t;
               while ( t-t0 < control_time )
                      {
                        sleep(Tstep - (t - tm));
                        curr_err = reading(pyrometer)-newT;
                        sigma    = sigma+Tstep*curr_err;
                        errtot   = errtot+Tstep*curr_err*curr_err;
                        delta    = (curr_err-prev_err)/Tstep;
                        TCsetp   = -(curr_err+sigma/Integ+delta*Deriv)*100/Prop;
                        if (TCsetp < TCmin) TCsetp = TCmin;
                        if (TCsetp > TCmax) TCsetp = TCmax;
                        set_temp(subs,TCsetp);
                        prev_err = curr_err;
                        tm = tm + Tstep;
                        tm=t;
                       }
                }
         echo(“ New substrate temp reached “, newT);
         Tsub_old = Tsub_new;
          }
/*          else   */
/*         {        */
/**/
/*=============================================================================================*/
/*======================  Maintain temperature ================================================*/
/*=============================================================================================*/
/**/
/*           echo(“ Maintaining temp at =”,Tsub_new);    */
/**/
          control_time = 5.0;                          /*   Control for 5 sec before    */
          newT         = Tsub_new;                     /*   testing for new Tsub        */
/*                   echo(“New temp:”,newT);     */
            sigma        = -setp(subs)*Integ*Prop/100;
          prev_err     = reading(pyrometer)-newT;
          set_ramp(subs,1000);
          tm           = t;
          t0           = t;
          while ( t-t0 < control_time )
                {
                sleep(Tstep - (t - tm));
                curr_err = reading(pyrometer)-newT;
/*                                      echo(“newT = “,newT);   */
                sigma    = sigma+Tstep*curr_err;
                errtot   = errtot+Tstep*curr_err*curr_err;
                delta    = (curr_err-prev_err)/Tstep;
                TCsetp   = -(curr_err+sigma/Integ+delta*Deriv)*100/Prop;
                    if (TCsetp < TCmin) TCsetp = TCmin;
```

```
                if (TCsetp > TCmax) TCsetp = TCmax;
               set_temp(subs,TCsetp);
               prev_err = curr_err;
               tm       = tm + Tstep;
               tm       = t;
               }
        }
    }
```

```
/**/
/*****************************************************************************/
/*                                                                           */
/*   This command file sets the substrate temperature during growth          */
/*                                    Stefan Svensson, ARL  Apr 21,2000      */
/*****************************************************************************/
/**/
Tsub_new   = 550;
tm         = t;
echo(“Tsub_new= “,Tsub_new);
```

```
/**/
/*****************************************************************************/
/*                                                                           */
/*   This command file kills the pyrometer control program Tsub.cmd          */
/*                                                                           */
/*                                    Stefan Svensson, ARL  May 24 2000      */
/*****************************************************************************/
/**/
/**/
#include <signal.h>
echo("pyro_id=  ",pyro_id);
kill(pyro_id,SIGKILL);
echo(" Pyrometer contro1 finished.");
```

```
/*********************************************************************/
/*                                                                   */
/*  This command procedure kills the data log                        */
/*  The data is stored in the file                                   */
/*  C:/u/mbe/recipes/stefan/log_data/outfile.dat                     */
/*                                                                   */
/*********************************************************************/

#include <signal.h>
echo("sigkill=",SIGKILL);
echo("log_id=",log_id);
kill(log_id,SIGKILL);
echo(" Stdlog1 finished. Data is in c:\\u\\mbe\\recipes\\stefan\\log_data\\outfile.dat");
```

# Distribution

Admnstr
Defns Techl Info Ctr
Attn DTIC-OCP
8725 John J Kingman Rd Ste 0944
FT Belvoir VA 22060-6218

DARPA
Attn S Welby
3701 N Fairfax Dr
Arlington VA 22203-1714

Ofc of the Secy of Defns
Attn ODDRE (R&AT)
The Pentagon
Washington DC 20301-3080

Ofc of the Secy of Defns
Attn OUSD(A&T)/ODDR&E(R) R J Trew
3080 Defense Pentagon
Washington DC 20301-7100

AMCOM MRDEC
Attn AMSMI-RD W C McCorkle
Redstone Arsenal AL 35898-5240

Dir for MANPRINT
Ofc of the Deputy Chief of Staff for Prsnnl
Attn J Hiller
The Pentagon Rm 2C733
Washington DC 20301-0300

SMC/CZA
2435 Vela Way Ste 1613
El Segundo CA 90245-5500

US Army ARDEC
Attn AMSTA-AR-TD M Fisette
Bldg 1
Picatinny Arsenal NJ 07806-5000

US Army Info Sys Engrg Cmnd
Attn AMSEL-IE-TD F Jenia
FT Huachuca AZ 85613-5300

US Army Natick RDEC Acting Techl Dir
Attn SBCN-T P Brandler
Natick MA 01760-5002

US Army Simulation, Train, & Instrmntn Cmnd
Attn AMSTI-CG M Macedonia
Attn J Stahl
12350 Research Parkway
Orlando FL 32826-3726

US Army Soldier & Biol Chem Cmnd Dir of Rsrch & Techlgy Dirctrt
Attn SMCCR-RS I G Resnick
Aberdeen Proving Ground MD 21010-5423

US Army Tank-Automtv Cmnd Rsrch, Dev, & Engrg Ctr
Attn AMSTA-TR J Chapin
Warren MI 48397-5000

US Army Train & Doctrine Cmnd
Battle Lab Integration & Techl Dirctrt
Attn ATCD-B
FT Monroe VA 23651-5850

US Military Academy
Mathematical Sci Ctr of Excellence
Attn MADN-MATH MAJ M Huber
Thayer Hall
West Point NY 10996-1786

Nav Surface Warfare Ctr
Attn Code B07 J Pennella
17320 Dahlgren Rd Bldg 1470 Rm 1101
Dahlgren VA 22448-5100

Hicks & Associates Inc
Attn G Singley III
1710 Goodrich Dr Ste 1300
McLean VA 22102

Palisades Inst for Rsrch Svc Inc
Attn E Carr
1745 Jefferson Davis Hwy Ste 500
Arlington VA 22202-3402

Director
US Army Rsrch Ofc
Attn AMSRL-RO-D JCI Chang
Attn AMSRL-RO-EN W D Bach
PO Box 12211
Research Triangle Park NC 27709

## Distribution (cont'd)

US Army Rsrch Lab
Attn AMSRL-DD J M Miller
Attn AMSRL-CI-AI-R Mail & Records Mgmt
Attn AMSRL-CI-AP Techl Pub (3 copies)
Attn AMSRL-CI-LL Techl Lib (3 copies)

US Army Rsrch Lab (cont'd)
Attn AMSRL-D D R Smith
Attn AMSRL-SE-RL S Svensson (4 copies)
Adelphi MD 20783-1197

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*



| 1. AGENCY USE ONLY *(Leave blank)* | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | October 2000 | Final, January–June 2000 |

**4. TITLE AND SUBTITLE** Procedure for Substrate Temperature Control Using the Pyrometer During MBE Growth

**5. FUNDING NUMBERS**

DA PR: AH94
PE: 62705A

**6. AUTHOR(S)** Stefan Svensson

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
U.S. Army Research Laboratory
Attn: AMSRL-SE-RL email: svensson@arl.army.mil
2800 Powder Mill Road
Adelphi, MD 20783-1197

**8. PERFORMING ORGANIZATION REPORT NUMBER**
ARL-MR-491

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
U.S. Army Research Laboratory
2800 Powder Mill Road
Adelphi, MD 20783-1197

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

**11. SUPPLEMENTARY NOTES**
ARL PR: 0NE6J2
AMS code: 622705.H94

**12a. DISTRIBUTION/AVAILABILITY STATEMENT** Approved for public release; distribution unlimited.

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** *(Maximum 200 words)*

Command procedures have been developed for the Army Research Laboratory molecular beam epitaxy (MBE) computer control system that allow a user to automatically outgas and desorb the oxide from substrates before growth, as well as set substrate temperatures based on pyrometer readings during growths. These procedures allow completely unattended growth of structures once suitable temperatures have been determined.

**14. SUBJECT TERMS** MBE, computer control, pyrometer

**15. NUMBER OF PAGES** 28

**16. PRICE CODE**

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| Unclassified | Unclassified | Unclassified | UL |

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102